Multi-core Curriculum Development at Georgia Tech: Experience and Future Steps

Ada Gavrilovska, Hsien-Hsin-Lee, Karsten Schwan, Sudha Yalamanchili, Matt Wolf

CERCS

Georgia Institute of Technology

Background for Multi-core

- CERCS team consists of CS and ECE faculty, awarded equipment and grant funding for multi-core curriculum development:

> Key Idea: "… to upgrade the core curriculum at the College of Computing (CoC) and School of Electrical and Computer Engineering (ECE) at Georgia Tech to better prepare future generations of hardware and software practitioners … to harness the potential of current and future multi-core platforms."

- Bring concurrency and parallelism back into CS/CompE education by coordinated updates to an entire ecosystem of courses

- Engage broad set of faculty beyond PIs

Single New Course – Not a Viable Solution

- At what level?
  - senior or graduate students – too late
  - freshman or sophomore – too early to address all concepts and challenges, and possibility of `turning off' video game generation' through potentially `dry' technical topics
- Administratively challenging to introduce new `required' course
- Limited coverage in terms of number of students exposed

GT Solution: Modules throughout Curriculum

Need for flexibility in multi-core offerings and content:

- Level of detail in experience and understanding of issues related to concurrency and multi-core platforms differs for students specializing in platforms and infrastructure (systems, architecture) vs. applications (high performance, pervasive-robotics, ...) vs. theory (design, modeling)
- CoC's new 'Threads' program in Computer Science education tailors distinct academic threads to match students interests
- Joint CoC/ECE efforts to create new joint MS curriculum can reach beyond traditional CS/ECE student population
- Outreach to non-CS/ECE faculty through efforts with Science and Engineering (e.g., CSE) and international cooperation (GT Lorraine, Korea program in embedded software, TUM, ETH Zurich)

"If it's worth saying once, it's worth saying at least three times." – received wisdom

I. Course Modules

- Develop course modules for insertion into a wide range of courses
  - undergraduate and graduate
  - CoC and ECE
  - systems and architecture (incl. verification)
  - compilers and programming languages
  - high performance computing and applications
  - enterprise computing and information integration
  - interface applications in embedded domain (e.g., robotics, vision)

II. Problem Sets, Software Stack, Web Repository

- Evolve Knowledge Base:
  - Multi-core web repository (including links to non-GT efforts)
  - Build and share knowledge about important tools and packages:
    - VTune, threads checker, threads, locking, concurrency and language techniques (including formal methods)
  - Create sample problem sets, projects, involving representative hardware
  - Extend to real-life applications (online data visualization, vision, robotics, information integration, ...)
- Build and maintain simulation and other software stacks (e.g., MPI/OpenMP, messaging, ...)
- Extend to low-level, hardware-near systems (e.g., hypervisors, comm. and graphics subsystems)

Course Modules – Some Detail

- Format:
  - Depending on class format, will include combination of 1-3 week lectures, with accompanying slides, handouts and reference material, homework(s) and laboratory assignment(s), or longer term (1/3 to full semester) class projects (same for all, or choice of topics)
- Content:
  - Will focus on new architectural enhancements, techniques for exploiting parallelism at the instruction and threads levels, operating systems design and implementation issues, shifts in programming and program compilation (e.g., multi-threaded programming), correctness issues, performance analysis and debugging, and application development.

Roadmap

- Introduce course modules in courses taught by faculty involved in proposal
  - Systems and Arch.: CS4210, CS6210, CS7210, ECE3055
  - initial upgrades already took place in Spring06 (lectures and/or labs), process continues…
- Expand module development to other 3xxx and higher level course
  - Compilers and Languages (CS4240) **and** Applications (CS 6230)
  - hire TAs from CS and ECE to interface between proposal Co-PIs and faculty teaching other affected courses, to provide, create, and adapt appropriate educational material, as well as help construct and maintain appropriate software stacks
- Initiate upgrades to entry-level courses
  - administrative limitations in ease of modifying 1xxx and 2xxx level courses due to large student enrollment, multiple course sections, several instructors; changes are regulated at College/Dept-level, but there is room for innovative efforts (e.g., UG seminar, ‘fun’ projects in graphics/robotics/embedded, ...)

List of Courses under Immediate Consideration

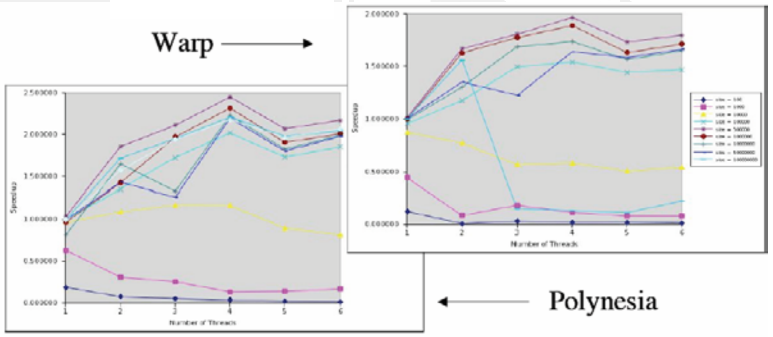
- CS3210 Design of Operating Systems
- CS4210 Advanced Operating Systems
- CS4290/CS6290 Advanced Computer Organization
- CS6210 Advanced Operating Systems
- CS6230 High Performance Parallel Computing: Tools and Applications
- CS7210 Distributed Computing
- CS8803HPC High Performance Communications
- ECE3055 Computer Architecture and Operating Systems
- ECE4100/ECE6100 Advanced Computer Architecture
- ECE6101 Parallel and Distributed Computer Architecture
- ECE7102 RISC Architecture
- CS 3220 Processor Design (Arch. and Verification)
- CS 4235/6235 Embedded and Real-time Systems

Case Study: CS4210 Advanced Operating Systems (Gavrilovska)

- Course heavily centered around threads and concurrency
  - two thirds of course discusses various concurrency related issues: threads and multithread programming, address space switches and IPC, webservers and RPC…
  - includes a sequence of three Pthreads-based projects where experimentation and performance analysis constitute 30% of project grade
- New additions:
  - discussion on trends in multi-core platforms
  - greater emphasis on select multithreading topics:
    - synchronization and deadlocks,
    - advanced synchronization constructs, and
    - scheduling – general algorithms, and scheduling for chip multiprocessors
- Planned additional updates:
  - ensure availability of single CPU, SMP, and dual-core systems for project development and evaluation, and require comparative performance analysis
    - currently only some students/groups
  - use of tools – VTune licenses pending

Case Study: CS6230 Intro to High Performance Computing (Wolf)

- Course is designed to be useful for both CS and non-CS graduate students
  - Deals extensively with design and application of parallel algorithms, methods and tools, and architecture.
  - Adaptation to new/novel hardware is a recurring theme



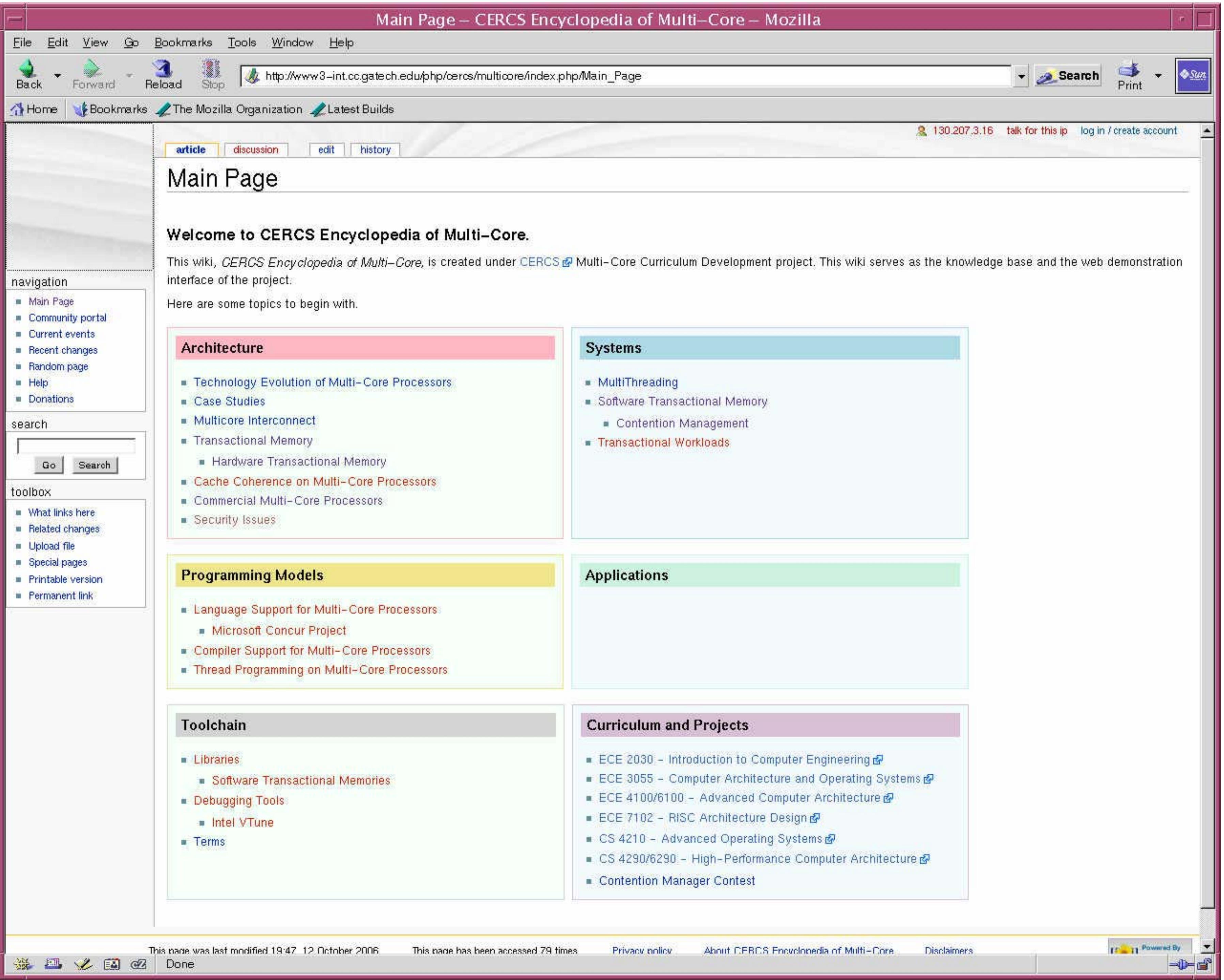
  - **Addition last semester**: Students were given a relatively open-ended assignment to explore trade-offs in multi-core & hyper-threading using SMP & multi-core hardware.
    - Analysis of performance of OpenMP – down to level of compiler instruction choices – was discussed w/student data
  - **Upcoming work**: Extend module w/use of performance tools; issues in multi-core + VT in HPC

Upcoming: ECE/CoC 8XXX Multicore Systems

- Conceived as an advanced, project intensive graduate course
  - Coupled with industry guest lectures
  - Industry influenced projects?
- Organized as 10-12 modules covering architecture (core and platform), programming and operating systems
- Taught by 3-5 faculty
- Goal: first offering Fall 2007

Current Infrastructure Support

- Hardware available for instruction
  - dual-core nodes in Netlab (remote boot, ...), dual SMP teaching facility, small number of larger MPs (8-way) machines, multi-core IXP platforms, …
- Environments and tools
  - primarily open source, based on Linux or other Unix flavors
    - (Windows in select courses or areas)
  - Linux kernel-based performance monitoring
  - Commercial (Intel) performance tools coming soon…
- Open web and knowledge repository


Main Page – CERCS Encyclopedia of Multi–Core – Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www3-int.cc.gatech.edu/php/cercs/multicore/index.php/Main_Page
Search Print
Home Bookmarks The Mozilla Organization Latest Builds
130.207.3.16 talk for this ip log in / create account
article discussion edit history
navigation
Main Page
Community portal
Current events
Recent changes
Random page
Help
Donations
search
Go Search
toolbox
What links here
Related changes
Upload file
Special pages
Printable version
Permanent link
Done

Main Page

Welcome to CERCS Encyclopedia of Multi–Core.

This wiki, *CERCS Encyclopedia of Multi–Core*, is created under CERCS Multi–Core Curriculum Development project. This wiki serves as the knowledge base and the web demonstration interface of the project.

Here are some topics to begin with.

Architecture

- Technology Evolution of Multi–Core Processors
- Case Studies
- Multicore Interconnect
- Transactional Memory
  - Hardware Transactional Memory
- Cache Coherence on Multi–Core Processors
- Commercial Multi–Core Processors
- Security Issues

Systems

- MultiThreading
- Software Transactional Memory
  - Contention Management
- Transactional Workloads

Programming Models

- Language Support for Multi–Core Processors
  - Microsoft Concur Project
- Compiler Support for Multi–Core Processors
- Thread Programming on Multi–Core Processors

Applications

Toolchain

- Libraries
  - Software Transactional Memories
- Debugging Tools
  - Intel VTune
- Terms

Curriculum and Projects

- ECE 2030 – Introduction to Computer Engineering
- ECE 3055 – Computer Architecture and Operating Systems
- ECE 4100/6100 – Advanced Computer Architecture
- ECE 7102 – RISC Architecture Design
- CS 4210 – Advanced Operating Systems
- CS 4290/6290 – High–Performance Computer Architecture
- Contention Manager Contest

This page was last modified 19:47 12 October 2006 This page has been accessed 79 times Privacy policy About CERCS Encyclopedia of Multi–Core Disclaimers

Multicore Curricular Infrastructure Goals

- Insight: why?
  - Understanding causality between HW and SW
    - prototype methodology developed jointly with Intel MTL
    - currently being integrated with GEMS modeling and simulation environment
- Experimentation: how?
  - Extensible interfaces to memory, interconnect, and I/O behaviors
    - construction of laboratory exercises to reinforce foundational classroom material
- Exploration: what if?
  - Integration of models for technology and architecture exploration
- Status: Considering modification to available tool chains, e.g., GEMS or basic SIMICS
  - target Fall 2007 for first use in graduate multicore course

Ties to Ongoing Research

- Students involved, for credit or for thesis work (UG, MS, PhD) in a range or research projects where concurrency and/or multicore issues are key elements
  - Scalable Hypervisors
  - Embedded and pervasive systems
  - Architecture
  - High performance computing and communications
    - Collaboration directly with ORNL, Sandia, ...
  - Middleware and enterprise systems